

An Efficient Synthesis Algorithm for Parametric Markov Chains Against Linear Time Properties*

Yong Li^{1,2}, Wanwei Liu³, Andrea Turrini¹, Ernst Moritz Hahn¹, and Lijun Zhang¹

¹ State Key Laboratory of Computer Science, Institute of Software, CAS, China

² University of Chinese Academy of Sciences, China

³ College of Computer Science, National University of Defense Technology, China

Abstract. In this paper, we propose an efficient algorithm for the parameter synthesis of PLTL formulas with respect to parametric Markov chains. The PLTL formula is translated to an almost fully partitioned Büchi automaton which is then composed with the parametric Markov chain. We then reduce the problem to solving an optimisation problem, allowing to decide the satisfaction of the formula using an SMT solver. The algorithm works also for interval Markov chains. The complexity is linear in the size of the Markov chain, and exponential in the size of the formula. We provide a prototype and show the efficiency of our approach on a number of benchmarks.

1 Introduction

Model checking, an automatic verification technique, has attracted much attention as it can be used to verify the correctness of software and hardware systems [1, 10, 12]. In classical model checking, temporal formulas are often used to express properties that one wants to check.

Probabilistic verification problems have been studied extensively in recent years. Markov chains (MCs) are a prominent probabilistic model used for modelling probabilistic systems. Properties are specified using probabilistic extensions of temporal logics such as probabilistic CTL (PCTL) [22] and probabilistic LTL (PLTL) [5] and their combination PCTL*. In the probabilistic setting, most of the observations about CTL and LTL carry over to their probabilistic counterpart. An exception is the complexity for verifying PLTL: here one could have a double exponential blowup. This is the case, because in general nondeterministic Büchi automata cannot be used directly to verify LTL properties, as they will cause imprecise probabilities in the product. In turn, it is often necessary to construct their deterministic counterparts in terms of other types of automata, for instance Rabin or Parity automata, which adds another exponential blowup. As a result, most of the work in literature focuses on branching time verification problems. Moreover, state-of-the-art tools such as PRISM [29] and MRMC [25] can handle large systems with PCTL specifications, but rather small systems –if at all– for PLTL specifications.

* We have very recently noticed the paper accepted at CAV [2] which treats the non-parametric Markov chains using a similar approach. Our approach has been developed in parallel.

In the seminal paper by Courcoubetis and Yannakakis [13], it is shown that for MCs the PLTL model checking problem is in PSPACE. They perform transformations of the Markov chain model recursively according to the LTL formula. At each step, the algorithm replaces a subformula rooted at a temporal operator with a newly introduced proposition; meanwhile, it refines the Markov chain with that proposition, and such refinement preserves the distribution. Then, it is finally boiled down to the probabilistic model checking upon a propositional formula. At the refinement step the state space is doubled, thus resulting in a PSPACE algorithm. Even if it is theoretically a single exponential algorithm for analysing MCs with respect to PLTL, it has not been exploited in the state-of-the-art probabilistic model checkers.

In automata-based approaches, one first translates the LTL formula into a Büchi automaton and then analyses the product of the MC and the Büchi automaton. This is sufficient for qualitative properties, i.e., to decide whether the specification is satisfied with probability 1. For quantitative properties, the Büchi automaton needs to be further transformed into a deterministic variant. Such a determinisation step usually exploits Safra’s determinisation construction [33]. Several improvements have been made in recent years, see for instance [30, 31, 34]. Model checkers such as PRISM [29] and LiQUOR [9] handle PLTL formulas by using off-the-shelf tools (e.g. (J)LTL2DSTAR [28]) to perform this determinisation step. To avoid the full complexity of the deterministic construction, Chatterjee et al. [7] have proposed an improved algorithm for translating the formulas of the FG-fragment of LTL to an extension of Rabin automata. Recently [18], this algorithm has been extended to the complete LTL. Despite the above improvements, the size of the resulting deterministic automaton is still the bottleneck of the approach for linear temporal properties. In [14], it is first observed that the second blowup can be circumvented by using *unambiguous Büchi automata* (UBAs) [6]. The resulting algorithm has the same complexity as the one in [13]. Despite the importance of probabilistic model checking, unfortunately, the algorithm in [14] is less recognised. To the best of the authors knowledge, it is not applied in any of the existing model checkers. Recently, in [27], the authors construct the so called *limit deterministic* Büchi automata that are exponential in the size of LTL\GU formula φ , which is another fragment of LTL. The approach is only applied to the analysis of qualitative PLTL of the form $\mathbb{P}_{>0}[\varphi]$.

In this paper, we present a further improvement of the solution proposed in [14], adapted directly to solving the parameter synthesis problem for parametric Markov chains. We exploit a simple construction translating the given LTL formula to a reverse deterministic UBA, and then build the product of the parametric Markov chains. We then extract an equation system from the product, then the synthesis problem reduces to the existence of a solution of the equation system. Further, we remark that the related interval Markov chains can be handled by our approach as well. We integrate our approach in the model checker ISCASMC [21], and employ SMT solver to solving the obtained equation system. We present detailed experimental results, and observe that our implementation can deal with some real-world probabilistic systems modelled by parametric Markov chains.

Related Work. In [20], they first use state elimination to compute the reachability probability for parametric Markov models. This has been improved by Dehnert *et al.* [17]. An-

other related models is interval Markov chains, which can be interpreted as a family of Markov chains [8,24] whose transition probabilities lie within the interval ranges. In [8], they also considered model checking ω -regular properties with interval Markov chains. They showed that the synthesis of interval Markov chains problem against ω -PCTL is decidable in PSPACE. In [24], they considered interval Markov chains as abstraction models by using three-valued abstraction for Markov chains. To our best knowledge, it is the first time that one can easily integrate parameter synthesis algorithm that is exponential in the size of LTL formulas over parametric Markov chains.

2 Preliminaries

Given a set W , we say that an infinite sequence $\varpi = w_0 w_1 \dots$ is an ω -word if $\varpi \in W^\omega$. Given a finite word $\nu = v_0 \dots v_k$ and a finite or infinite word $\varpi = w_0 w_1 \dots$, we denote by $\nu \cdot \varpi$ the *concatenation* of ν and ϖ , i.e., the finite or infinite word $\nu \cdot \varpi = v_0 \dots v_k w_0 w_1 \dots$, respectively. We may just write $\nu\varpi$ instead of $\nu \cdot \varpi$. We denote by $[1..n]$ the set of natural numbers $\{1, \dots, n\}$.

Probability Theory. A *measure* over a measurable space (Ω, \mathcal{F}) is a function $\mu: \mathcal{F} \rightarrow \mathbb{R}^{\geq 0}$ such that $\mu(\emptyset) = 0$ and, for each countable family $\{\Omega_i\}_{i \in I}$ of pairwise disjoint elements of \mathcal{F} , $\mu(\cup_{i \in I} \Omega_i) = \sum_{i \in I} \mu(\Omega_i)$. If $\mu(\Omega) \leq 1$, then we call μ a *sub-probability measure* and, if $\mu(\Omega) = 1$, then we call μ a *probability measure*. We say that μ is a *discrete* measure over Ω if \mathcal{F} is discrete. In this case, for each $X \subseteq \Omega$, $\mu(X) = \sum_{x \in X} \mu(\{x\})$ and we drop brackets whenever possible. For a set Ω , we denote by $\text{Disc}(\Omega)$ the set of discrete probability measures over Ω , and by $\text{SubDisc}(\Omega)$ the set of discrete sub-probability measures over Ω . We call $X \subseteq \Omega$ the *support* of a measure μ if $\mu(\Omega \setminus X) = 0$; in particular, if μ is discrete, we denote by $\text{Supp}(\mu)$ the minimum support set $\{x \in \Omega \mid \mu(x) > 0\}$. Moreover, we denote by δ_x , for $x \in \Omega$, the *Dirac* measure such that for each $X \subseteq \Omega$, $\delta_x(X) = 1$ if $x \in X$, 0 otherwise. If δ_x is discrete, then it holds that for each $y \in \Omega$, $\delta_x(y) = 1$ if $y = x$ and $\delta_x(y) = 0$ if $y \neq x$. In case Ω is countable, then the probability measure $\mu: \Omega \rightarrow [0, 1]$ over the discrete measurable space $(\Omega, 2^\Omega)$ can be obtained by imposing that $\sum_{x \in \Omega} \mu(x) = 1$; μ is also called a *probability distribution*.

Graph Theory. A *directed graph* G is a pair $G = (V, E)$ where V is a finite non-empty set of *vertices*, also called *nodes*, and $E \subseteq V \times V$ is the set of *edges* or *arcs*. Given an arc $e = (u, v)$, we call the vertex u the *head* of e , denoted by $\text{head}(e)$, and the vertex v the *tail* of e , denoted by $\text{tail}(e)$. In the remainder of the paper we consider only directed graphs and we refer to them just as graphs.

A *path* π is a sequence of edges $\pi = e_1 e_2 \dots e_n$ such that for each $1 \leq i < n$, $\text{tail}(e_i) = \text{head}(e_{i+1})$; we say that v is *reachable* from u if there exists a path $\pi = e_1 \dots e_n$ such that $\text{head}(e_1) = u$ and $\text{tail}(e_n) = v$.

A *strongly connected component* (SCC) is a set of vertices $C \subseteq V$ such that for each pair of vertices $u, v \in C$, u is reachable from v and v is reachable from u ; we say that a graph $G = (V, E)$ is *strongly connected* if V is an SCC. We say that an SCC C is *non-extensible* if for each SCC C' of G , $C \subseteq C'$ implies $C' = C$. Without loss of generality, in the remainder of this paper we consider only non-extensible SCCs.

We define the partial order \preceq over the SCCs of the graph G as follows: given two SCCs C_1 and C_2 , $C_1 \preceq C_2$ if there exist $v_1 \in C_1$ and $v_2 \in C_2$ such that v_2 is reachable from v_1 . We say that an SCC C is *maximal* with respect to \preceq if for each SCC C' of G , $C \preceq C'$ implies $C' = C$. We may call the maximal SCCs as *bottom SCCs*, BSCC for short.

A graph can be enriched with labels as follows: a *labelled graph* G is a triple $G = (V, \Sigma, E)$ where V is a finite non-empty set of *vertices*, Σ is a finite set of *labels*, and $E \subseteq V \times \Sigma \times V$ is the set of *labelled edges*. The notations and concepts on graphs trivially extend to labelled graphs.

Generalized Büchi Automata. A *generalized Büchi automaton* (GBA) \mathcal{A} is a tuple $\mathcal{A} = (\Sigma, Q, \mathbf{T}, Q_0, ACC)$ where Σ is a finite *alphabet*, Q is a finite set of *states*, $\mathbf{T}: Q \times \Sigma \rightarrow 2^Q$ is the *transition function*, $Q_0 \subseteq Q$ is the set of *initial states*, and $ACC = \{F_i \subseteq Q \mid i \in [1..k]\}$ is the set of *accepting sets*.

A *run* of \mathcal{A} over an infinite word $w = a_0a_1\ldots \in \Sigma^\omega$ is an infinite sequence $\sigma = q_0a_0q_1a_1q_2\ldots \in (Q \cdot \Sigma)^\omega$ such that $q_0 \in Q_0$ and for each $i \in \mathbb{N}$ it is $q_{i+1} \in \mathbf{T}(q_i, a_i)$. Similarly, a *run* of \mathcal{A} over a finite word $w = a_0a_1\ldots a_k \in \Sigma^*$ is a finite sequence $\sigma = q_0a_0q_1a_1q_2\ldots a_kq_{k+1} \in Q \cdot (\Sigma \cdot Q)^*$ such that $q_0 \in Q_0$ and for each $i \in \{0, \dots, k\}$ it is $q_{i+1} \in \mathbf{T}(q_i, a_i)$. Let $\text{Inf}(\sigma) = \{(q, a, q') \in \mathbf{T} \mid \forall i \in \mathbb{N}. \exists j \geq i. (q_j, a_j, q_{j+1}) = (q, a, q')\}$ be the set of tuples (q, a, q') occurring infinitely often in σ . The run σ is *accepting* if $\text{Inf}(\sigma) \cap F_i \neq \emptyset$ for each $i \in [1..k]$. The word w is accepted by \mathcal{A} if there is an accepting run of \mathcal{A} over w ; we denote by $\mathcal{L}(\mathcal{A})$ the *language* of \mathcal{A} , i.e., the set of infinite words accepted by \mathcal{A} .

Given the GBA $\mathcal{A} = (\Sigma, Q, \mathbf{T}, Q_0, ACC)$, for the sake of convenience, we denote by \mathcal{A}^q the GBA $\mathcal{A}^q = (\Sigma, Q, \mathbf{T}, \{q\}, ACC)$ with initial state q and accordingly for $U \subseteq Q$ we let $\mathcal{A}^U \stackrel{\text{def}}{=} (\Sigma, Q, \mathbf{T}, U, ACC)$.

The graph $G = (V, \Sigma, E)$ underlying a GBA \mathcal{A} is the graph whose set of vertices (nodes) V is the set of states S of \mathcal{A} and there is an edge $e \in E$ labelled with $a \in \Sigma$ from q to q' if $q' \in \mathbf{T}(q, a)$. In this case, we say that q is an *a-predecessor* of q' and q' is an *a-successor* of q .

Given a GBA \mathcal{A} , we say that

- \mathcal{A} is *deterministic*, if $|Q_0| = 1$ and $|\mathbf{T}(q, a)| = 1$ for each $q \in Q$ and $a \in \Sigma$;
- \mathcal{A} is *reverse deterministic* if each state has exactly one *a-predecessor* for each $a \in \Sigma$;
- \mathcal{A} is *unambiguous* if for each $q \in Q$, $a \in \Sigma$, and $q', q'' \in \mathbf{T}(q, a)$ such that $q' \neq q''$, we have $\mathcal{L}(\mathcal{A}^{q'}) \cap \mathcal{L}(\mathcal{A}^{q''}) = \emptyset$; and
- \mathcal{A} is *separated* if $\mathcal{L}(\mathcal{A}^q) \cap \mathcal{L}(\mathcal{A}^{q'}) = \emptyset$ for each pair of states $q, q' \in Q$, $q \neq q'$.

We say that a state $q \in Q$ is *reenterable* if q has some predecessor in \mathcal{A} . Let Q' be the set of all reenterable states of \mathcal{A} and consider the GBA $\mathcal{A}' = (\Sigma, Q', \mathbf{T}', Q', ACC')$ where $\mathbf{T}' = \mathbf{T}|_{Q' \times \Sigma}$ and $ACC' = \{F'_i = F_i \cap Q' \mid F_i \in ACC, i \in [1..k]\}$. Then, we say that \mathcal{A} is *almost unambiguous* (respectively *almost separated*, *almost reverse deterministic*) if \mathcal{A}' is unambiguous (respectively separated, reverse deterministic).

For an (almost) separated GBA \mathcal{A} , if for each $\alpha \in \Sigma^\omega$ there exists some state q of \mathcal{A} such that $\alpha \in \mathcal{L}(\mathcal{A}^q)$, then we say that \mathcal{A} is (almost) *fully partitioned*. Clearly, if

an automaton is (almost) fully partitioned, then it is also (almost) separated, (almost) unambiguous and (almost) reverse deterministic.

As an example of GBA that is reverse-deterministic and separated but not fully partitioned, consider the automaton \mathcal{A} in Fig. 1. The fact that \mathcal{A} is not fully partitioned is clear since no word starting with xw is accepted by any of the states q_1 , q_2 , or q_3 . One can easily check that \mathcal{A} is indeed reverse-deterministic but checking the separated property can be more involved. The

checks involving q_1 are trivial, as it is the only state enabling a transition with label x or w ; for the states q_2 and q_3 , the separated property implies that given any $w_1 \in \mathcal{L}(\mathcal{A}^{q_2})$, it is not possible to find some $w_2 \in \mathcal{L}(\mathcal{A}^{q_3})$ such that $w_1 = w_2$. For instance, suppose the number of the most front y 's in w_1 is odd, it must be the case that the most front y 's are directly followed by x or w . In order to match w_1 , we must choose y instead of z on transition (q_2, q_1) . It follows that the number of the most front y 's in w_2 is even. We can get similar result when the number of the most front y 's in w_1 is even. Thus w_1 and w_2 can never be the same.

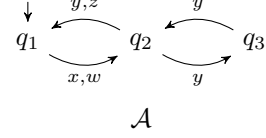


Fig. 1. An example of generalised Büchi automaton

3 Parametric Markov Chains and Probabilistic LTL

In this section we recall the definitions of parametric Markov chains as presented in [20], interval Markov chain considered in [3, 8, 24] and of the logic PLTL. In addition, we consider the translation of LTL formulas to GBAs which is used later for analysing PLTL properties.

3.1 Parametric Markov Chains

Before introducing the parametric Markov chain model, we briefly present some general notation. Given a finite set $V = \{x_1, \dots, x_n\}$ with domain in \mathbb{R} , an *evaluation* v is a partial function $v: V \rightarrow \mathbb{R}$. Let $\text{Dom}(v)$ denote the domain of v ; we say that v is total if $\text{Dom}(v) = V$. A *polynomial* p over V is a sum of monomials $p(x_1, \dots, x_n) = \sum_{i_1, \dots, i_n} a_{i_1, \dots, i_n} \cdot x_1^{i_1} \cdots x_n^{i_n}$ where each $i_j \in \mathbb{N}$ and each $a_{i_1, \dots, i_n} \in \mathbb{R}$. A *rational function* f over V is a fraction $f(x_1, \dots, x_n) = \frac{p_1(x_1, \dots, x_n)}{p_2(x_1, \dots, x_n)}$ of two polynomials p_1 and p_2 over V ; we denote by \mathcal{F}_V the set of all rational functions over V . Given $f \in \mathcal{F}_V$, $V' \subseteq V$, and an evaluation v , we let $f[V'/v]$ denote the rational function obtained by replacing each occurrence of $v \in V' \cap \text{Dom}(v)$ with $v(v)$.

Definition 1. A parametric Markov chain (PMC), is a tuple $\mathcal{M} = (S, L, \bar{s}, V, \mathbf{P})$ where S is a finite set of states, $L: S \rightarrow \Sigma$ is a labelling function where Σ is a finite set of state labels, $\bar{s} \in S$ is the initial state, V is a finite set of parameters, and $\mathbf{P}: S \times S \rightarrow \mathcal{F}_V$ is a transition matrix.

We now define the PMC induced with respect to a given evaluation:

Definition 2. Given a PMC $\mathcal{M} = (S, L, \bar{s}, V, \mathbf{P})$ and an evaluation v , the PMC \mathcal{M}_v induced by v is the tuple $\mathcal{M}_v = (S, L, \bar{s}, V \setminus \text{Dom}(v), \mathbf{P}_v)$ where the transition matrix $\mathbf{P}_v: S \times S \rightarrow \mathcal{F}_{V \setminus \text{Dom}(v)}$ is given by $\mathbf{P}_v(s, t) = \mathbf{P}(s, t)[\text{Dom}(v)/v]$.

We say that a *total* evaluation is *well-defined* for a PMC \mathcal{M} if $\mathbf{P}_v(s, s') \in [0, 1]$ and $\sum_{t \in S} \mathbf{P}_v(s, t) = 1$ for each $s, s' \in S$. In the remainder of the paper we consider only well-defined evaluations and we require that, for a given PMC \mathcal{M} and two states $s, t \in S$, if $\mathbf{P}_v(s, t) > 0$ for some evaluation v , then $\mathbf{P}_{v'}(s, t) > 0$ for the v' considered. We may omit the actual evaluation v when we are not interested in the actual value for $\mathbf{P}_v(s, t)$, such as for the case $\mathbf{P}_v(s, t) > 0$.

We use $|S|$ to denote the number of states, and $|\mathcal{M}|$ for the number of non-zero probabilistic transitions, i.e., $|\mathcal{M}| = |\{(s, s') \in S \times S \mid \mathbf{P}(s, s') > 0\}|$.

The *underlying graph* of a PMC \mathcal{M} is the graph $G = (V, E)$ where $V = S$ and $E = \{(s, s') \in S \times S \mid \mathbf{P}(s, s') > 0\}$.

A *path* is a sequence of states $\pi = s_0 s_1 \dots$ satisfying $\mathbf{P}(s_i, s_{i+1}) > 0$ for all $i \geq 0$. We call a path π *finite* or *infinite* if the sequence π is finite or infinite, respectively. We use $\pi(i)$ to denote the suffix $s_i s_{i+1} \dots$ and we denote by $\text{Paths}^{\mathcal{M}}$ and $\text{Paths}_{\text{fin}}^{\mathcal{M}}$ the set of all infinite and finite paths of \mathcal{M} , respectively. An infinite path $\pi = s_0 s_1 \dots$ defines the ω -word $w_0 w_1 \dots \in \Sigma^\omega$ such that $w_i = L(s_i)$ for $i \in \mathbb{N}$.

For a finite path $s_0 s_1 \dots s_k$, we denote by $\text{Cyl}(s_0 s_1 \dots s_k)$ the *cylinder set* of $s_0 s_1 \dots s_k$, i.e., the set of infinite paths starting with prefix $s_0 s_1 \dots s_k$. Given an evaluation v , we define the measure of the cylinder set by $\mathfrak{P}^{\mathcal{M}_v}(\text{Cyl}(s_0 s_1 \dots s_k)) \stackrel{\text{def}}{=} \delta_{\bar{s}}(s_0) \cdot \prod_{i=0}^{k-1} \mathbf{P}_v(s_i, s_{i+1})$. For a given PMC \mathcal{M} and an evaluation v , we can extend $\mathfrak{P}^{\mathcal{M}_v}$ uniquely to a probability measure over the σ -field generated by cylinder sets [26].

We call the bottom SCCs of the underlying graph G *ergodic sets* and for each ergodic set C , we call each state $s \in C$ *ergodic*. A nice property of a bottom SCC C is the so-called *ergodicity property*: for each $s \in C$, s will be reached again in the future with probability 1 from any state $s' \in C$, including s itself. Moreover, for each finite path π within C , π will be performed again in the future with probability 1.

In this paper we are particularly interested in ω -regular properties $\mathcal{L} \subseteq \Sigma^\omega$ and the probability $\mathfrak{P}^{\mathcal{M}}(\mathcal{L})$ for some measurable set \mathcal{L} . Such properties are known to be measurable in the σ -field generated by cylinders [35]. We write $\mathfrak{P}_s^{\mathcal{M}}$ to denote the probability function when assuming that s is the initial state of the PMC \mathcal{M} . To simplify the notation, we omit the superscript \mathcal{M} whenever \mathcal{M} is clear from the context and we use Π as a synonym for Paths .

3.2 Interval Markov chain

In this section we recall the definition of interval Markov chain [8, 24] and show how it can be converted to a parametric Markov chain.

Definition 3. An interval Markov chain (IMC) is a tuple $\mathcal{M} = (S, L, \bar{s}, \mathbf{P}_l, \mathbf{P}_u)$ where S , L and \bar{s} are as for PMCs while $\mathbf{P}_l, \mathbf{P}_u: S \times S \rightarrow [0, 1]$ are the transition matrices such that for each $s, s' \in S$, $\mathbf{P}_l(s, s') \leq \mathbf{P}_u(s, s')$.

We show how to convert an IMC to a PMC in the following. Given an IMC $\mathcal{M} = (S, L, \bar{s}, \mathbf{P}_l, \mathbf{P}_u)$, we define the corresponding PMC $\mathcal{M}' = (S, L, \bar{s}, \mathbf{P})$ as follows. For every pair of states, say (s, t) , we add a new parameter p_{st} to V such that $V = \{p_{st} \mid \mathbf{P}_l(s, t) \leq p_{st} \leq \mathbf{P}_u(s, t)\}$; then, we define \mathbf{P} as $\mathbf{P}(s, t) = p_{st}$. For instance, suppose in an IMC, there is a state s with two successors, namely t and w , with $\mathbf{P}_l(s, t) = 0.2$,

$\mathbf{P}_l(s, w) = 0.3$, $\mathbf{P}_u(s, t) = 0.7$ and $\mathbf{P}_u(s, w) = 0.5$. We add two parameters p_{st} and p_{sw} for the pairs (s, t) and (s, w) whose ranges are $[0.2, 0.7]$ and $[0.3, 0.5]$ respectively. Moreover, in order to get an instance of Markov chain from the resulting PMC, we must make sure that $p_{st} + p_{sw} = 1$.

3.3 Probabilistic Linear Time Temporal

Throughout the whole paper, we will assume that the state space S of any PMC is always equipped with labels that identify distinguishing state properties. For this, we let AP denote a set of atomic propositions. We assume $\Sigma = 2^{AP}$ as state labels, so that $L(s)$ specifies the subset of atomic propositions holding in state s .

We first recall the linear time temporal logic (LTL). The syntax of LTL is given by:

$$\varphi \stackrel{\text{def}}{=} p \mid \neg\varphi \mid \varphi \wedge \varphi \mid \mathbf{X}\varphi \mid \varphi \mathbf{U}\varphi$$

where $p \in AP$. We use standard derived operators, such as: $\varphi_1 \vee \varphi_2 \stackrel{\text{def}}{=} \neg(\neg\varphi_1 \wedge \neg\varphi_2)$, $\text{true} \stackrel{\text{def}}{=} a \vee \neg a$, $\varphi_1 \rightarrow \varphi_2 \stackrel{\text{def}}{=} \neg\varphi_1 \vee \varphi_2$, $\Diamond\varphi \stackrel{\text{def}}{=} \text{true}\mathbf{U}\varphi$, and $\Box\varphi \stackrel{\text{def}}{=} \neg(\Diamond\neg\varphi)$. Semantics is standard and is omitted here.

A probabilistic LTL (PLTL) formula has the form $\mathbb{P}_J(\varphi)$ where $J \subseteq [0, 1]$ is a non-empty interval with rational bounds and φ is an LTL formula. In a PMC \mathcal{M} with evaluation v , for a state $s \in S$ and a formula $\mathbb{P}_J(\varphi)$, we have:

$$s \models \mathbb{P}_J(\varphi) \text{ if and only if } \mathfrak{P}_s^{\mathcal{M}v}(\{\pi \in \Pi \mid \pi \models \varphi\}) \in J \quad (1)$$

where $\mathfrak{P}_s^{\mathcal{M}v}(\{\pi \in \Pi \mid \pi \models \varphi\})$, or $\mathfrak{P}_s^{\mathcal{M}v}(\varphi)$ for short, denotes the probability measure of the set of all paths which satisfy φ . The synthesis problem of this paper is thus to find such a v if possible or to prove that the LTL formula is invalid for all valid v . From the measurability of ω -regular properties, we can easily show that for any PLTL path formula φ , the set $\{\pi \in \Pi \mid \pi \models \varphi\}$ is measurable in the σ -field generated by the cylinder sets.

3.4 From LTL to Büchi automaton

The following section describes how we can transform a given LTL formula into a GBA which has the required properties for the subsequent model checking procedure.

Definition 4. *The set of elementary formulas $el(\varphi)$ for a given LTL formula φ is defined recursively as follows: $el(p) = \emptyset$ if $p \in AP$; $el(\neg\psi) = el(\psi)$; $el(\varphi_1 \wedge \varphi_2) = el(\varphi_1) \cup el(\varphi_2)$; $el(\mathbf{X}\psi) = \{\mathbf{X}\psi\} \cup el(\psi)$; and $el(\varphi_1 \mathbf{U}\varphi_2) = \{\mathbf{X}(\varphi_1 \mathbf{U}\varphi_2)\} \cup el(\varphi_1) \cup el(\varphi_2)$.*

Given a set $V \subseteq el(\varphi)$ and $a \in \Sigma = 2^{AP}$, we inductively define the satisfaction relation \Vdash for each subformula of φ as follows:

$$\begin{aligned} (V, a) \Vdash p & \quad \text{if } p \in a \text{ in the case of } p \in AP, \\ (V, a) \Vdash \neg\psi & \quad \text{if it is not the case that } (V, a) \Vdash \psi, \\ (V, a) \Vdash \varphi_1 \wedge \varphi_2 & \text{ if } (V, a) \Vdash \varphi_1 \text{ and } (V, a) \Vdash \varphi_2, \\ (V, a) \Vdash \mathbf{X}\psi & \quad \text{if } \mathbf{X}\psi \in V, \text{ and} \\ (V, a) \Vdash \varphi_1 \mathbf{U}\varphi_2 & \text{ if } (V, a) \Vdash \varphi_2 \text{ or } (V, a) \Vdash \varphi_1 \text{ and } (V, a) \Vdash \mathbf{X}(\varphi_1 \mathbf{U}\varphi_2). \end{aligned}$$

Finally, $\mathcal{A}_\varphi = (\Sigma = 2^{AP}, Q_\varphi, \mathbf{T}_\varphi, \{\varphi\}, ACC_\varphi)$ is the Büchi automaton where:

- $Q_\varphi = \{\varphi\} \cup 2^{el(\varphi)}$;
- $\mathbf{T}_\varphi(\varphi, a) = \{V \subseteq el(\varphi) \mid (V, a) \models \varphi\}$ and for each $V \subseteq el(\varphi)$, we have:
 $\mathbf{T}_\varphi(V, a) = \{U \subseteq el(\varphi) \mid \forall X\psi \in el(\varphi). X\psi \in V \iff (U, a) \models \psi\}$; and
- $ACC_\varphi = \{F_\psi\}$ where for each subformula $\psi = \varphi_1 U \varphi_2$ of φ , $F_\psi = \{(U, a, V) \in \mathbf{T}_\varphi \mid (V, a) \models \varphi_2 \text{ or } (V, a) \models \neg\psi\}$.

In Definition 4, each formula in $el(\varphi)$ is guaranteed to be of the form $X\varphi'$; the size of $el(\varphi)$ is precisely the number of temporal operators (i.e., X and U) occurring in φ .

Theorem 1 (cf. [11]). *For the automaton \mathcal{A}_φ , the following holds:*

1. *For each infinite word $\pi \in \Sigma^\omega$, we have $\pi \models \varphi$ if and only if $\pi \in \mathcal{L}(\mathcal{A}_\varphi)$.*
2. *More generally, for each $U \subseteq el(\varphi)$ and $X\psi \in el(\varphi)$ we have: $\pi \models \psi$ if and only if $X\psi \in U$, for every $\pi \in \mathcal{L}(\mathcal{A}_\varphi^U)$.*

It follows directly that

Corollary 1. *For each $U, V \subseteq el(\varphi)$, if $U \neq V$ then $\mathcal{L}(\mathcal{A}_\varphi^U) \cap \mathcal{L}(\mathcal{A}_\varphi^V) = \emptyset$. Moreover, \mathcal{A}_φ is both almost unambiguous and almost separated.*

We observe that for each subset $U \subseteq el(\varphi)$ and each $a \in \Sigma$, there is exactly one a -predecessor of U , namely the set $\{X\psi \in el(\varphi) \mid (U, a) \models \psi\}$. Hence, we also have the following conclusion.

Corollary 2. *The automaton \mathcal{A}_φ is almost reverse deterministic and fully partitioned.*

Intuitively, for any $w \in \Sigma^\omega$, we can find a state $U = \{X\psi \in el(\varphi) \mid w \models \psi\}$ and we observe that $w \in \mathcal{L}(\mathcal{A}_\varphi^U)$ by Theorem 1. Since \mathcal{A}_φ is already almost separated, it follows that it is also almost fully partitioned. Note that because of the non-reenterable initial state, the automaton may not be fully partitioned, but is almost fully partitioned.

4 Parameter Synthesis Algorithm

We consider a parametric Markov chain \mathcal{M} and an almost fully partitioned automaton $\mathcal{A} = (\Sigma, Q, \mathbf{T}, Q_0, ACC)$ obtained from the LTL specification, where $\mathcal{L}(\mathcal{A}^{q_1}) \cap \mathcal{L}(\mathcal{A}^{q_2}) = \emptyset$ if $q_1, q_2 \in Q_0$ and $q_1 \neq q_2$. To simplify the notation, in the following we assume that for a given PMC \mathcal{M} we have $S = \Sigma$ and $L(s) = s$ for each $s \in S$; this modification does not change the complexity of probabilistic model checking [13]. Below we define the product graph:

Definition 5. *Given the automaton $\mathcal{A} = (\Sigma, Q, \mathbf{T}, Q_0, ACC)$ and the PMC $\mathcal{M} = (S, L, \bar{s}, V, \mathbf{P})$, the product graph of \mathcal{A} and \mathcal{M} , denoted $\mathcal{G} = \mathcal{A} \times \mathcal{M}$, is the graph $\mathcal{G} = (\Gamma, \Delta)$ where $\Gamma = \{(q, s) \mid q \in Q, s \in S\}$ and $((q, s), (q', s')) \in \Delta$ (also written $(q, s) \Delta (q', s')$) if and only if $\mathbf{P}(s, s') > 0$ and $q' \in \mathbf{T}(q, L(s))$.*

Suppose that $ACC = \{F_1, \dots, F_k\}$. We say that an SCC C of \mathcal{G} is accepting if for each $F_i \in ACC$, there exist $(q, s), (q', s') \in C$ such that $(q, s) \Delta (q', s')$ and $(q, a, q') \in F_i$ for some $a \in \Sigma$.

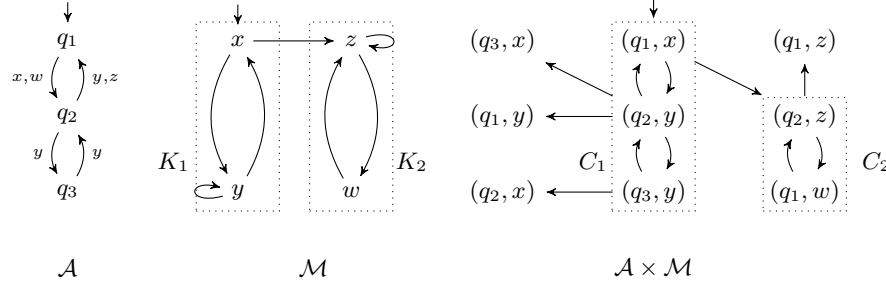


Fig. 2. The GBA \mathcal{A} from Fig. 1, a PMC \mathcal{M} , and their product $\mathcal{A} \times \mathcal{M}$

Given an SCC C of \mathcal{G} , we denote by $\mathcal{H}(C)$ the *corresponding SCC* of \mathcal{M} , where $\mathcal{H}(C) = \{s \in S \mid (q, s) \in C\}$. We denote by Proj a function to get the corresponding path of \mathcal{M} from the path of \mathcal{G} , i.e., $\text{Proj}((q_0, s_0)(q_1, s_1) \dots) = s_0 s_1 \dots$ and we usually call the path $s_0 s_1 \dots$ the *projection* of $(q_0, s_0)(q_1, s_1) \dots$. For convenience, we also write $\alpha \preceq \beta$ if the (finite) path α is a fragment of the (finite) path β .

Definition 6 (Complete SCC). For an SCC C of \mathcal{G} and $K = \mathcal{H}(C)$ the corresponding SCC of \mathcal{M} , we say that C is *complete* if for each finite path σ_K in K , we can find a finite path σ_C in C such that $\sigma_K = \text{Proj}(\sigma_C)$.

Consider the product $\mathcal{A} \times \mathcal{M}$ shown in Fig. 2. It has two non-trivial SCCs, namely C_1 and C_2 . Clearly, K_1 and K_2 are the corresponding SCCs of C_1 and C_2 , respectively. We observe that C_1 is a complete SCC while C_2 is not complete since the finite path zz of K_2 is not a projection of any finite path in C_2 . The key observation is that some transitions in the SCCs of \mathcal{M} may be lost in building the product, so we only consider the complete SCCs in the product to make sure that no transitions of the projection SCCs are missing.

The following lemma characterises the property of a complete SCC of \mathcal{G} .

Lemma 1. Consider a complete SCC C of \mathcal{G} with $\mathcal{H}(C) = K$ and an arbitrary finite path ρ_C in C . Then, there exists some finite path σ_K in K with the following property: for each finite path σ_C in C with $\sigma_K = \text{Proj}(\sigma_C)$, σ_C contains ρ_C as a fragment, i.e., $\rho_C \preceq \sigma_C$.

Proof. Clearly, C must be the product of K and some component of \mathcal{A} . Recall that \mathcal{A} is almost reverse-deterministic, then for each path α_C in C and each finite path β_K in K , there is at most one path $\gamma_C \cdot \alpha_C$ in C with $\text{Proj}(\gamma_C) = \beta_K$. In the following, we call $\gamma_C \cdot \alpha_C$ the β_K -backward-extension of α_C .

Given a finite path α in C , we define $\Gamma_\alpha(\gamma) \stackrel{\text{def}}{=} \{\beta \mid \text{Proj}(\beta) = \text{Proj}(\gamma), \alpha \not\preceq \beta\}$ for every finite path γ in C .

Then, for the path ρ_C , we define a sequence of finite paths $\rho_0, \rho_1, \rho_2, \dots$ as follows:

- $\rho_0 = \rho_C$. According to the definition, $\rho_0 \notin \Gamma_{\rho_C}(\rho_0)$, because $\rho_C \preceq \rho_C = \rho_0$. Moreover, we have that $\Gamma_{\rho_C}(\rho_0)$ is a finite set, let $\ell = |\Gamma_{\rho_C}(\rho_0)| \leq |Q|^n$, where n is the number of states along the finite path ρ_0 and Q is the state space of \mathcal{A} .

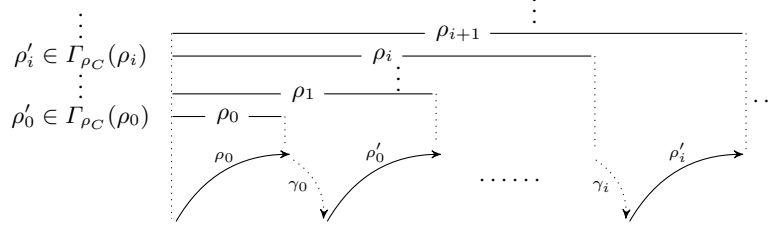


Fig. 3. Construction of ρ_t in Lemma 1

- For each $i \geq 0$, we impose the inductive hypothesis that $\rho_C \trianglelefteq \rho_i$. Then, if $|\Gamma_{\rho_C}(\rho_i)| > 0$, we arbitrarily choose a path $\rho'_i \in \Gamma_{\rho_C}(\rho_i)$. Since C is an SCC, then there exists some path γ_i connecting the last node of ρ_i to the first state of ρ'_i .
- Let $\rho_{i+1} = \rho_i \cdot \gamma_i \cdot \rho'_i$, then it is a finite path in C and $\rho_C \trianglelefteq \rho_{i+1}$. It is immediate to see that $|\Gamma_{\rho_C}(\rho_{i+1})| < |\Gamma_{\rho_C}(\rho_i)|$ since the set $\Gamma_{\rho_C}(\rho_{i+1})$ just consists of the $\text{Proj}(\rho_i \cdot \gamma_i)$ -backward-extensions of paths in $\Gamma_{\rho_C}(\rho_i)$. As we have mentioned, each path has at most one such extension, and the extension for ρ'_i , namely ρ_{i+1} , does not belong to $\Gamma_{\rho_C}(\rho_{i+1})$ since it involves the fragment ρ_C .

Therefore, there must exist $t \leq \ell$ such that $|\Gamma_{\rho_C}(\rho_t)| = 0$. Let $\sigma_K = \text{Proj}(\rho_t)$; then each finite path in C with projection σ_K must contain the fragment ρ_C . \square

For example, consider the product $\mathcal{A} \times \mathcal{M}$ from Fig. 2. Given the complete SCC C_1 and $\rho_C = (q_3, y)(q_2, y)$, we find the specific σ_K as follows.

1. $\rho_0 = \rho_C$. First we get $\Gamma_{\rho_C}(\rho_0) = \{(q_2, y)(q_3, y)\}$. Since $|\Gamma_{\rho_C}(\rho_0)| = 1 > 0$, we choose $\gamma_0 = (q_2, y)(q_1, x)(q_2, y)$ and for ρ'_0 , we have no choices but $(q_2, y)(q_3, y)$.
2. $\rho_1 = \rho_0 \cdot \gamma_0 \cdot \rho'_0$. Similarly, we need to compute $\Gamma_{\rho_C}(\rho_1)$. Since $\text{Proj}(\rho_1) = yyxyxy$ and one needs to visit x after traversing yy , which is impossible to bypass ρ_C to make it happen. Then it gives us $|\Gamma_{\rho_C}(\rho_1)| = 0$. Therefore, we set $\sigma_K = yyxyxy$ to be the specific finite path of K_1 .

Based on Lemma 1, the following corollary relates the paths in the product and the projected Markov chains:

Corollary 3. *Let C be a complete SCC of \mathcal{G} and $K = \mathcal{H}(C)$; consider two infinite paths σ_C in C and σ_K in K such that $\sigma_K = \text{Proj}(\sigma_C)$; let P_C and P_K be the following properties:*

- P_C : σ_C visits each finite path in C infinitely often;
- P_K : σ_K visits each finite path of K infinitely often.

Then P_C holds if and only if P_K holds.

The proof of Corollary 3 can be found in the appendix.

Definition 7. *We say that the SCC C of \mathcal{G} is locally positive if:*

1. *C is accepting and complete.*

2. $\mathcal{H}(C)$ is maximal with respect to $\preceq_{\mathcal{M}}$ (it is a so-called bottom SCC).

Consider again the example from Fig. 2. Assume the acceptance condition of \mathcal{A} is $ACC = \{\{(q_2, y, q_1)\}\}$; we observe that the SCC C_1 is both accepting and complete but not locally positive since $\mathcal{H}(C_1) = \{x, y\}$ is not a bottom SCC in \mathcal{M} .

According to Corollary 3, the ergodicity property of Markov chains, and the definition of Büchi acceptance, we have the following result.

Proposition 1. $\mathfrak{P}^{\mathcal{M}}(\mathcal{L}(\mathcal{A})) \neq 0$ if and only if there exists some locally positive SCC in \mathcal{G} .

For a given SCC, in order to decide whether it is locally positive, we have to judge whether it is complete. In general, doing so is a nontrivial task. Thanks to [13, Lemma 5.10], completeness can be checked efficiently:

Lemma 2. If \mathcal{A} is (almost) reverse deterministic, then the following two conditions are equivalent:

- i) C is complete, i.e., each finite path of $\mathcal{H}(C)$ is a projection of some finite path in C .
- ii) There is no other SCC C' of \mathcal{G} with $\mathcal{H}(C') = \mathcal{H}(C)$ such that $C' \preceq C$.

Intuitively, in the product \mathcal{G} composed by an almost reverse deterministic automaton \mathcal{A} and a PMC \mathcal{M} , the complete SCCs must be the SCCs whose preceded SCCs should not have the same projections. The detailed proof can be found in the appendix.

We now turn to the problem of computing the exact probability.

Theorem 2. Given a PMC \mathcal{M} and a fully partitioned Büchi automaton \mathcal{A} , let $\mathcal{G} = \mathcal{A} \times \mathcal{M}$ be their product. Let $\text{pos}(\mathcal{G})$ be the set of all locally positive SCCs of \mathcal{G} and $\text{neg}(\mathcal{G})$ be the set of all BSCCs of \mathcal{G} which are not locally positive. Further, for an SCC C let $C_{\mathcal{M}} = \{s \in S \mid \exists q \in Q. (q, s) \in C\}$ denote the set of states of \mathcal{M} occurring in C . We define the following equation system:

$$\mu(q, s) = \sum_{s' \in S} \left(\mathbf{P}(s, s') \cdot \sum_{(q, s) \in \Delta(q', s')} \mu(q', s') \right) \quad \forall q \in Q, s \in S \quad (2)$$

$$\sum_{\substack{q \in Q \\ (q, s) \in C}} \mu(q, s) = 1 \quad \forall C \in \text{pos}(\mathcal{G}), s \in C_{\mathcal{M}} \quad (3)$$

$$\mu(q, s) = 0 \quad \forall C \in \text{neg}(\mathcal{G}) \text{ and } (q, s) \in C \quad (4)$$

Then, it holds that $\mathfrak{P}^{\mathcal{M}_v}(\mathcal{L}(\mathcal{A})) = \sum_{q_0 \in Q_0} \mu(q_0, \bar{s})$ for any well-defined evaluation v .

In general, all locally positive SCCs can be projected to the BSCCs in the induced MC \mathcal{M}_v . In the original MC \mathcal{M}_v , the reachability probability of every state in the accepting BSCC should be 1. Thus in a locally positive SCC of \mathcal{G} , the probability mass 1 distributes on the states in which they share the same second component, i.e., s from state (q, s) . This follows from the fact that the resulting product \mathcal{G} is almost fully partitioned so that the probability is also partitioned.

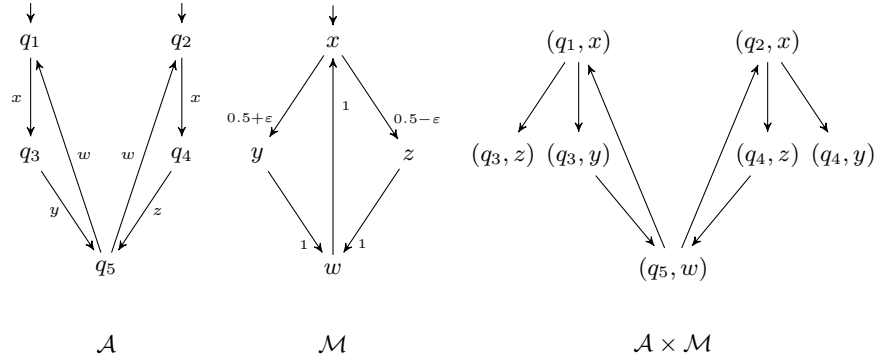


Fig. 4. An example of a GBA \mathcal{A} , a PMC \mathcal{M} , and their product $\mathcal{A} \times \mathcal{M}$

Example 1. Consider the PMC \mathcal{M} and the automaton \mathcal{A} as depicted in Fig. 4, together with their product graph. For clarity, we have omitted the isolated vertices like (q_1, w) and (q_5, y) , i.e., the vertices with no incoming or outgoing edges. One may check that \mathcal{A} is indeed separated, unambiguous, and reverse deterministic. The product of \mathcal{M} and \mathcal{A} consists of a single locally positive SCC $C = \{(q_1, x), (q_2, x), (q_3, y), (q_4, z), (q_5, w)\}$. We state the relevant part of the equation system resulting from equations (2) and (3) of Theorem 2:

$$\begin{aligned}
 \mu(q_3, z) &= 0 \\
 \mu(q_4, y) &= 0 \\
 \mu(q_1, x) &= (0.5 + \varepsilon) \cdot \mu(q_3, y) + (0.5 - \varepsilon) \cdot \mu(q_3, z) & \mu(q_1, x) + \mu(q_2, x) &= 1 \\
 \mu(q_2, x) &= (0.5 - \varepsilon) \cdot \mu(q_4, z) + (0.5 + \varepsilon) \cdot \mu(q_4, y) & \mu(q_3, y) &= 1 \\
 \mu(q_3, y) &= 1 \cdot \mu(q_5, w) & \mu(q_4, z) &= 1 \\
 \mu(q_4, z) &= 1 \cdot \mu(q_5, w) & \mu(q_5, w) &= 1 \\
 \mu(q_5, w) &= 1 \cdot (\mu(q_1, x) + \mu(q_2, x))
 \end{aligned}$$

(2)
(3)

We remark that the values for the nodes (q_3, z) and (q_4, y) as well as all isolated nodes like (q_1, w) or (q_5, y) are 0, because for them the inner summation in (2) is over the empty set. The family of solutions of this equation system has as non-zero values $\mu(q_1, x) = 0.5 + \varepsilon$, $\mu(q_2, x) = 0.5 - \varepsilon$, $\mu(q_3, y) = 1$, $\mu(q_4, z) = 1$, and $\mu(q_5, w) = 1$. From this, we have $\mathfrak{P}^{\mathcal{M}}(\mathcal{L}(\mathcal{A})) = \mu(q_1, x) + \mu(q_2, x) = 1$ for any well-defined evaluation v , i.e., an evaluation such that $v(\varepsilon) \in (-0.5, 0.5)$.

Let us summarise this section with the following results: given a parametric Markov chain \mathcal{M} and an LTL formula φ , both the emptiness checking and the quantitative-correctness computation could be done within time $\mathcal{O}(|\mathcal{M}| \cdot 2^{|\text{el}(\varphi)|})$. When the specification is given as an almost fully partitioned automaton \mathcal{A} , the time complexity is $\mathcal{O}(|\mathcal{M}| \cdot |\mathcal{A}|)$.

Table 1. Experimental Results for Parametric Markov chain models

Model (Constants) Property	Constants	$ S_{\mathcal{M}} $	$ V_{\mathcal{G}} $	$SCC_{\mathcal{G}}$	SCC_{pos}	$T_{\mathcal{G}}$	$ Vars_{Z3} $	$ Cons_{Z3} $	T_{Z3}	T_{mc}
Crowds (TotalRuns, CrowdSize)	2,50	20534	20535	1372	1	<1	314	320	13	13
	2,60	28446	28447	1938	1	<1	374	380	42	43
	2,70	39131	39132	2613	1	<1	434	440	80	82
	2,80	51516	51517	3388	1	1	494	500	92	94
	2,90	65601	65602	4263	1	1	554	560	162	164
	2,100	81386	81387	5238	1	2	614	620	198	202
$\mathbb{P} \geq 0.9[\text{GF}(\text{newInstance} \wedge \text{runCount} = 0 \wedge \text{observe} \geq 1)]$	2,110	98871	98872	6313	1	2	674	680	404	407
	2,120	118056	118057	7488	1	2	734	740	436	439
BRP (N, MAX)	512,80	1084476	1085505	1111	0	15	NE	NE	NE	16
	512,100	1351476	1352505	1131	0	22	NE	NE	NE	22
	512,120	1618476	1619505	1151	0	41	NE	NE	NE	42
	512,140	1885476	1886505	1171	0	54	NE	NE	NE	54
	512,160	2152476	2153505	1191	0	75	NE	NE	NE	76
	512,180	2419476	2420505	1211	0	87	NE	NE	NE	87
$\mathbb{P} \geq 0.9[\text{GF}(s = 5 \wedge T)]$	512,200	2686476	2687505	1231	0	110	NE	NE	NE	111
	512,220	2953476	2954505	1251	0	136	NE	NE	NE	137
RPSC (N, L)	5,15	300030	305492	1	0	5	NE	NE	NE	5
	5,20	402430	409442	1	0	9	NE	NE	NE	9
	5,25	504830	513392	1	0	12	NE	NE	NE	12
	5,30	607230	617342	1	0	19	NE	NE	NE	19
	5,35	709630	721292	1	0	25	NE	NE	NE	25
	5,40	812030	825242	1	0	32	NE	NE	NE	32
$\mathbb{P} \geq 0.9[\text{GF}(\neg \text{know}A \wedge \text{know}B)]$	5,45	914430	929192	1	0	39	NE	NE	NE	39
	5,50	1016830	1033142	1	0	46	NE	NE	NE	47

Remark 1. We note that in [3], a single exponential algorithm is presented for model checking PLTL properties on Markov chains. However, the approach presented there contains a flaw, which has been fixed in a subsequent report [4]. In this fix, the authors have also exploited UBAs. Comparing to our approaches, their corrected version has a higher complexity. In particular, they reduce computing the probability of a language being accepted by an UBA to solving a system of linear equations encoding the probability of the languages accepted by a non-deterministic finite automaton. The resulting complexity is indeed polynomial in the size of the UBA and of the Markov chain, but it is higher than the one of the algorithm we consider in this work: for instance, it involves checking accepting states which is cubic in the size of the system, whereas our approach is linear. Moreover, the fix was not implemented in their tool tulip [3].

5 Experiment Results

We have implemented our synthesis algorithm for LTL properties of parametric Markov chains in our tool ISCASMC using an explicit state-space representation. The machine we used for the experiments is a 3.6 GHz Intel Core i7-4790 with 16 GB 1600 MHz DDR3 RAM of which 12 GB assigned to the tool; the timeout has been set to 30 minutes. We considered three models, namely the Bounded Retransmission Protocol (BRP) [23] in the version of [15], Randomized Protocol for Signing Contracts (RPSC) [19] and the Crowds protocol for anonymity [32]. We have replaced the probabilistic choices by parametric choices to obtain the admissible failure probabilities.

In our implementation, we use Z3 [16] to solve the equation system of Theorem 2 because by using parametric transition probabilities the equation system from Theorem 2 becomes nonlinear. Performance results of the experiments are shown in Tables 1

Table 2. Experimental Results for Parametric Markov chain models

Model (Constants) Property	Constants	$ S_{\mathcal{M}} $	$ V_{\mathcal{G}} $	$SCC_{\mathcal{G}}$	SCC_{pos}	$T_{\mathcal{G}}$	$ Vars_{Z3} $	$ Cons_{Z3} $	T_{Z3}	T_{mc}
Crowds (TotalRuns, CrowdSize)	2,50	20534	82149	1374	0	4	NE	NE	NE	5
	2,60	28446	113797	1940	0	6	NE	NE	NE	7
	2,70	39131	156537	2615	0	10	NE	NE	NE	11
	2,80	51516	206077	3390	0	21	NE	NE	NE	21
	2,90	65601	262417	4265	0	26	NE	NE	NE	27
	2,100	81386	325557	5240	0	108	NE	NE	NE	108
	2,110	98871	395497	6315	0	109	NE	NE	NE	110
	2,120	118056	472237	7490	0	118	NE	NE	NE	118
BRP (N, MAX)	512,10	149976	893661	1055	0	32	NE	NE	NE	33
	512,20	283476	1692861	1075	0	114	NE	NE	NE	114
	512,30	416976	2492061	1095	0	241	NE	NE	NE	241
	512,40	550476	3291261	1115	0	476	NE	NE	NE	477
	512,50	683976	4090461	1135	0	622	NE	NE	NE	623
	512,60	817476	4889661	1155	0	962	NE	NE	NE	962
	512,70	950976	5688861	1175	0	1219	NE	NE	NE	1220
	512,80	1084476	MO	–	–	–	–	–	–	–
RPSC (N, L)	5,15	300030	1200117	4	1	68	300031	300036	32	106
	5,20	402430	1609717	4	1	122	402431	402436	42	169
	5,25	504830	2019317	4	1	178	504831	504836	49	230
	5,30	607230	2428917	4	1	278	607231	607236	75	373
	5,35	709630	2838517	4	1	384	709631	709636	86	488
	5,40	812030	3248117	4	1	490	812031	812036	95	600
	5,45	914430	3657717	4	1	606	914431	914436	104	719
	5,50	1016830	4067317	4	1	726	1016831	1016836	108	839

to 3, where the columns have the following meaning: In column “Model”, the information about the model, the name of the constants influencing the model size, and the analysed property is provided; “Constants” contains the values for the constants defining the model; “ $|S_{\mathcal{M}}|$ ” and “ $|V_{\mathcal{G}}|$ ” denote the number of states and vertices in \mathcal{M} and \mathcal{G} , respectively; “ $SCC_{\mathcal{G}}$ ” reports the number of non-trivial SCCs checked in \mathcal{G} out of which “ SCC_{pos} ” are the positive ones; and “ $T_{\mathcal{G}}$ ” is the time spent by constructing and checking the product graph; “ $|Vars_{Z3}|$ ”, “ $|Cons_{Z3}|$ ” and “ T_{Z3} ” record the number of variables and constraints of the equation system we input into Z3 and its solution time; and “ T_{mc} ” gives the total time spent for constructing and analysing the product \mathcal{G} and solving the equation system. In the tables, entries marked by “NE” mean that Z3 has not been executed as there were no locally positive SCCs, thus the construction and evaluation of the equation system can be avoided; entries marked by “–” mean that the operation has not been performed since the analysis has been interrupted in a previous stage; the marks “TO” and “MO” stand for a termination by timeout or memory out.

As we can see from Tables 1 and 2 relative to parametric Markov Chains, the implementation of the analysis method presented in this paper is able to check models in the order of millions of states. The model checking time is mainly depending on the behavior of Z3 and on the number and size of the SCCs of the product, since each SCC has to be classified as positive or negative; regarding Z3, we can see that the time it requires for solving the provided system is loosely related to the size of the system: the Crowds cases in Table 1 take hundreds of seconds for a system of size less than one thousand while the RPSC cases in Table 2 are completed in less than one hundred seconds even if the system size is more than one million. In Table 3, we list some experimental results for the Crowds protocol modelled as an interval Markov chain. As for the previous cases, it is the solution of the equation system to limit the size of the models we can

Table 3. Experimental Results for Crowds Interval Markov chain model

Model (Constants) Property	Constants	$ S_{\mathcal{M}} $	$ V_{\mathcal{G}} $	$SCC_{\mathcal{G}}$	SCC_{pos}	$T_{\mathcal{G}}$	$ Vars_{Z3} $	$ Cons_{Z3} $	T_{Z3}	T_{mc}
Crowds (TotalRuns, CrowdSize)	2,6	423	424	36	1	<1	240	473	<1	<1
	2,8	698	699	55	1	<1	380	760	<1	<1
$\mathbb{P} \geq 0.9[\text{GF}(\text{newInstance} \wedge \text{runCount} = 0 \wedge \text{observe} \geq 1)]$	2,10	1041	1042	78	1	<1	552	1115	1	1
	2,12	1452	1453	105	1	<1	756	1538	15	15
	2,14	1931	1932	136	1	<1	992	2029	15	15
	2,16	3093	3094	210	1	<1	1260	2588	TO	TO

analyse, so a more performing solver would improve considerably the applicability of our approach, in particular when we can not exclude that the formula is satisfiable, as happens when there are no positive SCCs.

6 Conclusion

In this paper we have surveyed the parameter synthesis of PLTL formulas with respect to parametric Markov chains. The algorithm first transforms the LTL specification to an almost fully partitioned automaton and then builds the product graph of the model under consideration and this automaton. Afterwards, we reduce the model checking problem to solving an (nonlinear) equation system, which allows us employ an SMT solver to obtain feasible parameter values. We have conducted experiments to demonstrate that our techniques indeed work for models of realistic size. To the best of our knowledge, our method is the first approach for the PLTL synthesis problem for parametric Markov chains which is single exponential in the size of the property.

References

1. C. Baier and J.-P. Katoen. *Principles of Model Checking*. MIT Press, 2008.
2. C. Baier, S. Kiefer, J. Klein, S. Klüppelholz, D. Müller, and J. Worrell. Markov chains and unambiguous Büchi automata. In *CAV*, 2016.
3. M. Benedikt, R. Lenhardt, and J. Worrell. LTL model checking of interval Markov chains. In *TACAS*, volume 7795 of *LNCS*, pages 32–46, 2013.
4. M. Benedikt, R. Lenhardt, and J. Worrell. Model checking Markov chains against unambiguous Büchi automata. CoRR, available at <http://arxiv.org/abs/1405.4560>, 2014.
5. A. Bianco and L. de Alfaro. Model checking of probabilistic and nondeterministic systems. In *FSTTCS*, volume 1026 of *LNCS*, pages 499–513, 1995.
6. O. Carton and M. Michel. Unambiguous Büchi automata. *TCS*, 297(1-3):37–81, 2003.
7. K. Chatterjee, A. Gaiser, and J. Kretínský. Automata with generalized Rabin pairs for probabilistic model checking and LTL synthesis. In *CAV*, volume 8044 of *LNCS*, pages 559–575, 2013.
8. K. Chatterjee, K. Sen, and T. A. Henzinger. Model-checking ω -regular properties of interval Markov chains, 2008.
9. F. Ciesinski and C. Baier. LiQuor: A tool for qualitative and quantitative linear time analysis of reactive systems. In *QEST*, pages 131–132, 2006.
10. E. M. Clarke. The birth of model checking. In *25 Years of Model Checking*, volume 5000 of *LNCS*, pages 1–26, 2008.

11. E. M. Clarke, O. Grumberg, and K. Hamaguchi. Another look at LTL model checking. In *CAV*, volume 818 of *LNCS*, pages 415–427, 1994.
12. E. M. Clarke, O. Grumberg, and D. Peled. *Model Checking*. MIT Press, 2001.
13. C. Courcoubetis and M. Yannakakis. The complexity of probabilistic verification. *J. of the ACM*, 42(4):857–907, 1995.
14. J.-M. Couvreur, N. Saheb, and G. Sutre. An optimal automata approach to LTL model checking of probabilistic systems. In *LPAR*, volume 2850 of *LNCS*, pages 361–375, 2003.
15. P. D’Argenio, B. Jeannet, H. Jensen, and K. Larsen. Reachability analysis of probabilistic systems by successive refinements. In *PAPM/PROBMIV*, volume 2165 of *LNCS*, pages 39–56, 2001.
16. L. De Moura and N. Bjørner. Z3: An efficient smt solver. In *TACAS, TACAS’08/ETAPS’08*, pages 337–340, Berlin, Heidelberg, 2008. Springer-Verlag.
17. C. Dehnert, S. Junges, N. Jansen, F. Corzilius, M. Volk, H. Bruintjes, J.-P. Katoen, and E. Ábrahám. PROPhESY: A PRObabilistic ParamEter SYNthesis Tool. In *CAV*, pages 214–231. Springer International Publishing, 2015.
18. J. Esparza and J. Kretínský. From LTL to Deterministic Automata: A Safrless Compositional Approach. In *CAV*, volume 8559 of *LNCS*, pages 192–208. Springer International Publishing, 2014.
19. S. Even, O. Goldreich, and A. Lempel. A randomized protocol for signing contracts. *Commun. ACM*, 28(6):637–647, June 1985.
20. E. M. Hahn, H. Hermanns, and L. Zhang. Probabilistic reachability for parametric Markov models. *STTT*, 13(1):3–19, 2011.
21. E. M. Hahn, Y. Li, S. Schewe, A. Turrini, and L. Zhang. IscasMC: A web-based probabilistic model checker. In *FM*, volume 8442 of *LNCS*, pages 312–317, 2014.
22. H. Hansson and B. Jonsson. A logic for reasoning about time and reliability. *FAC*, 6(5):512–535, 1994.
23. L. Helmink, M. Sellink, and F. Vaandrager. Proof-checking a data link protocol. In *TYPES*, volume 806 of *LNCS*, pages 127–165, 1994.
24. J.-P. Katoen, D. Klink, M. Leucker, and V. Wolf. Three-valued abstraction for probabilistic systems. *The Journal of Logic and Algebraic Programming*, 81(4):356 – 389, 2012.
25. J.-P. Katoen, I. S. Zapreev, E. M. Hahn, H. Hermanns, and D. N. Jansen. The ins and outs of the probabilistic model checker MRMC. *Performance Evaluation*, 68(2):90–104, 2011.
26. J. G. Kemeny, J. L. Snell, and A. W. Knapp. *Denumerable Markov Chains*. D. Van Nostrand Company, 1966.
27. D. Kini and M. Viswanathan. Limit Deterministic and Probabilistic Automata for LTL\GU. In *TACAS*, volume 9035 of *LNCS*, pages 628–642, 2015.
28. J. Klein and C. Baier. On-the-fly stuttering in the construction of deterministic ω -automata. In *CIAA*, volume 4783 of *LNCS*, pages 51–61, 2007.
29. M. Kwiatkowska, G. Norman, and D. Parker. PRISM 4.0: Verification of probabilistic real-time systems. In *CAV*, volume 6806 of *LNCS*, pages 585–591, 2011.
30. W. Liu and J. Wang. A tighter analysis of Piterman’s Büchi determinization. *Information Processing Letters*, 109(16):941–945, 2009.
31. N. Piterman. From nondeterministic Büchi and Streett automata to deterministic parity automata. *LMCS*, 3(3:5), 2007.
32. M. Reiter and A. Rubin. Crowds: Anonymity for web transactions. *ACM TISSEC*, 1(1):66–92, 1998.
33. S. Safra. On the complexity of ω -automata. In *FOCS*, pages 319–327, 1988.
34. S. Schewe. Tighter bounds for the determinisation of Büchi automata. In *FoSSaCS*, volume 5504 of *LNCS*, pages 167–181, 2009.
35. M. Y. Vardi. Automatic verification of probabilistic concurrent finite-state programs. In *FOCS*, pages 327–338, 1985.

Appendix

A Proof of Theorem 1

Here we only give the proof of the second claim. we denote by $fst(\varpi)$ the *first* element of ϖ , i.e, $fst(\varpi) = w_0$.

Proof. According to the definition of transition function \mathbf{T}_φ , we just need to prove $(V, fst(\pi)) \Vdash \psi$ if and only if $\pi \models \psi$ for every $\pi \in \mathcal{L}(\mathcal{A}_\varphi^U)$, where $V \in \mathbf{T}_\varphi(U, fst(\pi))$. Following we show the claim by induction on the structure of formula ψ .

1. $\psi = p \in AP$. By satisfaction relation \Vdash , $(V, fst(\pi)) \Vdash \psi$ is equivalent to $p \in fst(\pi)$ and further $\pi \models p$.
2. $\psi = \neg\psi_1$ or $\psi = \psi_1 \wedge \psi_2$. Trivial by direct application of induction hypotheses.
3. $\psi = X\psi_1$. Assume $\pi \models X\psi_1$ and $\pi \in \mathcal{L}(\mathcal{A}_\varphi^U)$, we have $\pi(1) \in \mathcal{L}(\mathcal{A}_\varphi^V)$ and $\pi(1) \models \psi_1$. By induction hypothesis $X\psi_1 \in V$ if and only if $\pi(1) \models \psi_1$ for every $\pi(1) \in \mathcal{L}(\mathcal{A}_\varphi^V)$. Moreover, $X\psi_1 \in V$ is equivalent to $(V, fst(\pi)) \Vdash X\psi_1$, which completes the proof.
4. $\psi = \psi_1 U \psi_2$. We divide it into two cases:
 - (a) $\pi \models \psi_2$. Trivial by induction hypothesis.
 - (b) $\pi \models \psi_1 \wedge X(\psi_1 U \psi_2)$. By using above induction hypotheses for both ψ_1 and $X(\psi_1 U \psi_2)$, we complete the proof.

B Proof of Corollary 3

Proof. Suppose σ_C and its projection σ_K is depicted as follows.

$$\begin{array}{ccccccc} \sigma_C & (q_0, s_0) & \longrightarrow & (q_1, s_1) & \longrightarrow & (q_2, s_2) & \longrightarrow \cdots \longrightarrow (q_i, s_i) \longrightarrow \cdots \\ \sigma_K & s_0 & \longrightarrow & s_1 & \longrightarrow & s_2 & \longrightarrow \cdots \longrightarrow s_i \longrightarrow \cdots \end{array}$$

1. $P_C \implies P_K$. Take any finite path σ'_k of K , we can always get a finite path σ'_c of C such that $\sigma'_k = \text{Proj}(\sigma'_c)$ since C is complete. By assumption, σ_C visits σ'_c infinitely often and $\sigma_K = \text{Proj}(\sigma_C)$, then σ_K must visit σ'_k infinitely often.
2. $P_K \implies P_C$. Take any finite path σ'_c of C . By Lemma 1, we can find some finite path ρ_k in K (not necessarily that $\rho_k = \text{Proj}(\sigma'_c)$ holds) such that with all ρ_c of C and $\rho_k = \text{Proj}(\rho_c)$, we have $\sigma'_c \preceq \rho_c$. According to the assumption, σ_K visits ρ_k infinitely often and $\sigma_K = \text{Proj}(\sigma_C)$. It must be the case that σ_C visits some above finite path ρ_c infinitely often and ρ_c exists since C is complete.

C Proof of Lemma 2

Proof. We show it by contradiction in the following.

1. $i) \Rightarrow ii)$. Suppose C is complete and there exists an SCC C' such that $\mathcal{H}(C') = \mathcal{H}(C)$ and there exists an arc from a state (q_0, s) of C' to a state (q_1, t) of C . Since \mathcal{G} is reverse deterministic, the number of $L(s)$ -predecessors of (q_1, t) is at most one, thus the arc (s, t) of $\mathcal{H}(C)$ does not have a corresponding arc in C . Therefore $i)$ does not hold.
2. $ii) \Rightarrow i)$. Suppose C is an SCC and there exists some arc (s, t) in $\mathcal{H}(C)$ which does not have a corresponding arc in C . We first find a state, say (q, t) in C , and get its immediate $L(s)$ -predecessor (p, s) , which is obviously not in C since \mathcal{G} is reverse-deterministic. We assume that C' is the SCC contains state (p, s) with $\mathcal{H}(C') = \mathcal{H}(C)$, which leads to $C' \preceq C$. Thus we complete the proof.